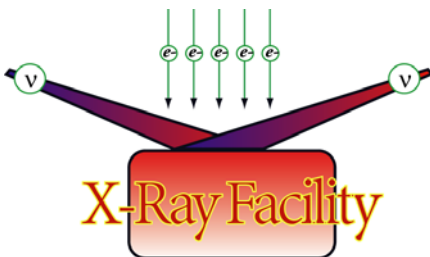


# Soma's Computer Notes

## Search & Replace using *find* and *rpl*



*Procedure for searching and replacing  
strings in multiple files across multiple  
directories*

Table of contents

**Introduction** ..... 3

**UNIX commands: *find, exec, & sed***..... 3

    Web-search ..... 3

        Link1: ..... 3

        Link2 ..... 3

**UNIX commands: *rpl*** ..... 5

    Get and Install Rpl ..... 5

    Using rpl to replace multiple .html file locations ..... 5

**Conclusion**..... 5

© 2000-09 Thayumanasamy Somasundaram  
414KLB, Institute of Molecular Biophysics  
91 Chieftan Way, Florida State University,  
Tallahassee, FL 32306-4380  
E-mail: [tsomasundaram@fsu.edu](mailto:tsomasundaram@fsu.edu) • URL: <http://www.sb.fsu.edu/~soma>  
Phone 850.644.6448 • Fax 850.644.7244  
August 14, 2009  
*Logos, Figures & Photos © of the respective Instrument Manufacturers*

# Search and Replace Using *find* and *rp1*

## *Procedure for searching and replacing strings in multiple files across multiple directories*

Version: 2; August 14, 2009;

### Introduction

This note is intended to help the [X-Ray Facility](#) (XRF) users search and replace `string1` by `string2` that occurs in many places in multiple files across multiple directories using the Linux/UNIX commands `find`, `exec` and `sed`. Copy of this Note will be posted in [XRF Resources page](#) shortly after receiving suggestions and corrections from the users. This note updated on August 14, 2009 to include `rp1` and was first written in November 29, 2007.

### UNIX commands: *find*, *exec*, & *sed*

`Find`, `exec`, & `sed` are UNIX/Linux commands that can be utilized to do routine work with ease. The problem I needed to solve was to replace `string1` by `string2` across multiple files across multiple directories quickly. A quick search of the Internet yielded some clues about how to proceed.

#### Web-search

TSomasundaram | November 06, 2007 | Search and replace several lines in several files in several directories. Google search term: [linux replace recursive](#)

**Link1:** [tips.webdesign10.com/recursively-find-and-replace-linux](http://tips.webdesign10.com/recursively-find-and-replace-linux)

[Recursively Find and Replace in GNU/Linux](#) | 2007, February 6 - 11:42pm — WebDesign10

Web designers often link to `index.html` in directories throughout a Web site — or even worse, only partially throughout a Web site. If you are dealing with a static HTML site, it should be fairly easy to fix with this recipe.

The following line in the GNU/Linux terminal will find and replace (delete) the text `index.html` recursively in all files, starting in the current directory:

```
find ./ * -type f -exec sed -i 's/index.html//g' {} \;
```

**Link2:** [www.jonasblog.com/2006/05/search-and-replace-in-all-files-within-a-directory-recursively.html](http://www.jonasblog.com/2006/05/search-and-replace-in-all-files-within-a-directory-recursively.html)

So, to search recursively through directories, looking in all the files for a particular string, and to replace that string with something else (on Linux) the following command should work:

```
find ./ -type f -exec sed -i 's/string1/string2/' {} \;
```

Where `string1` is the search and `string2` is the replacement.

Then I started my own trials and took some help from Michael Zawrotny, IMB System Manager. What I wanted to do was to update an old URL that was part of the template to a new location. Since this URL was found in almost all `.html` files for [www.sb.fsu.edu/~soma](http://www.sb.fsu.edu/~soma) and [www.sb.fsu.edu/~xray](http://www.sb.fsu.edu/~xray), I needed first to back-up the old html files so that I will not lose my webpages. Then I tested the code with `grep` rather than `sed`. This way I will see whether code was working before implementing it. Then I tried in on a sub-directory before doing it in the whole site.

```
1) find ./ -type f -exec grep -i '~webguide' {} \;
```

**Explanation:** Here we are using the `find` command with `-type f` option to get only *files* and NOT *directories*. Then we are using `-exec` option of `find` with `grep` as an operator. The `grep` command has option `-i` and is looking for a pattern `'~webguide'`. Then we have couple of symbols that are part of `exec` command `{}` `\;`. Note the combination of curly braces, an empty space, forward slash, and a semi-colon. This combination has to be written exactly as shown (**red-arrows** indicate empty space).

```
-exec grep -i 'webguide' {} \;
```

```
2) find ./ * -exec grep -i -H '\-2005' {} \; | more
```

**Explanation:** Here we are using the `find` command with wild card. Then we are using `-exec` option of `find` with `grep` as an operator. The `grep` command has option `-i` and `-H` options looking for a pattern `'\-2005'`. What I am looking for is actually `'-2005'`, but since `'-'` is a special character, I have to escape it with `'\'`, the escape character back-ward slash. I am also using `-H` option to get the filenames under `grep`. Once again, note the combination of curly braces, an empty space, forward slash, and a semi-colon. This combination has to be written exactly as shown (**red-arrows** indicate empty space).

```
3) find ./ -iname \*.htm\* -exec grep -i "\-2005" {} \;
```

**Explanation:** Here we are using the `find` command with `-iname \*.htm\*` option. Then `'\'` is escape character and the wildcard combination. `'.htm\*` is to capture all `.htm`, and `.html` files and escaping the special character `*`. Rest of the command is same as before.

```
4) find ./ -iname \*.htm\* -exec sed -i 's/\-2005/\-2007/g' {} \;
```

**Explanation:** Here we are using the `find` command with `-iname \*.htm\*` option. Then `'\'` is escape character and the wildcard combination both in front and back of `'.htm'` is to capture all `.htm` and `.html` files. Then using `sed` to replace all `'-2005'` by `'-2007'`. Note the special format for `sed` and used with `-i` option where `s` stands for substitute and `g` stands for global (all occurrences), and `/string1/ /string2/` `delimiter` `string2` is substituted for `string1`. Once again we have to escape 'the dash' in front of 2005 with escape character (`'\'`)

```
5) find ./ -iname \*.htm\* -exec grep -i '~webguide' {} \;
```

**Explanation:** Here we are simply checking to make sure all the replacements have been done. The above command should give no output if everything has worked as planned.

## UNIX commands: *rpl*

UNIX/Linux command `find` can be utilized to do routine work with ease when the strings to be replaced `string1` by `string2` are small. But when the strings are long I found `rpl` to be more useful. `Rpl` can be used to replace strings across multiple files across multiple directories as well.

### Get and Install `Rpl`

`Rpl` utility may not be part of the original distribution and so one may not find it in all systems. So one has to get it and install it. For RedHat system I found the `rpm` in the following location: <http://www.laffeycomputer.com/rpl.html>

If one has Debian or Ubuntu system then you get the `rpl` utility using `apt-get` and install like the following:

```
soma@xyz:~$ which rpl
bash: type: rpl: not found
soma@xyz:~$ sudo apt-get install rpl
... ..
Unpacking rpl (from ../archives/rpl_1.5.5-1_all.deb) ...
Setting up rpl (1.5.5-1) ...
soma@xyz:~$ which rpl
rpl is /usr/bin/rpl
```

### Using `rpl` to replace multiple `.html` file locations

Once `rpl` is installed one can replace long strings on multiple locations, on multiple files, on multiple directories. I needed to replace a location for all my `.html` files with new location on all those files (since the old location was no longer valid). I went about first running `rpl` in verbose and simulation mode. After making sure everything is OK, I went ahead and replaced the locations. Things went well quickly.

```
1) rpl -vsRd -x '.html' 'www.site.com/loc/file' 'www.site.us/~ab/loc2' *
```

**Explanation:** Here we are using the `rpl` command with `-vsRd` and `-x '.html'` options to get only *files* that have extension `*.html`. The option `-v` means verbose mode, `-s` means we running the simulation (and not actually doing the replacing yet), `-R` means we are using the recursive mode (looking for files in `./` and its child directories, and `-d` means we are keeping the original time of creation (maintaining old time-stamp). Here `'www.site.com/loc/file'` is the original string that needs to be replace by `'www.site.us/~ab/loc2'`, and finally `'*'` means we are doing on all files (redundant) since we are specifying `'-x .html'`.

After making sure everything is OK, I went ahead and replaced the locations. Things went well quickly.

```
2) rpl -vRd -x '.html' 'www.site.com/loc/file' 'www.site.us/~ab/loc2' *
```

**Explanation:** Here we are using the `rpl` command with `-vRd` and `-x '.html'` option meaning we are no longer using simulation mode but actually doing the replacement.

## Conclusion

I hope this write-up is useful to everyone. Please send your comments to [Somasundaram](mailto:Somasundaram).